Approximate Dynamic Programming for lateral transshipment problems in multi-location inventory systems

Joern Meissner

Kühne Logistics University, Hamburg, Germany. joe@meiss.com,[*]

Olga V. Rusyaeva

Kühne Logistics University, Hamburg, Germany. olga.rusyaeva@the-klu.org

March 30, 2016

Abstract

Companies commonly allocate their inventories across multiple locations based on their historical sales rates. However, random fluctuations in customer purchases, such as those caused by weather conditions and other external factors, might cause significant deviations from expected demand, leading to excess stock in some locations and stockouts in others. To fix this mismatch, companies often turn to lateral transshipments, e.g., the movement of stock between locations of the same echelon.

In this paper, we examine multi-location inventory systems under periodic review with multiple opportunities for proactive transshipments within one order cycle. If stockouts occur, demand is lost with no opportunity to backorder. The objective of our model is to find an optimal policy that indicates the sources and the destinations of transshipments as well as the number of units, to maximise the profit of the network. We create a dynamic program that can, in principal, be solved to optimality using Bellman's equation. However, the size of the state and decision spaces makes it impossible to find the optimal policy for real-world sized problem instances. Thereby, we use forward approximate dynamic programming to find a near-optimal transshipment policy.

Finally, we conduct an extensive numerical study to gauge the performance of our transshipment policy. For small size instances, we compare our policy to the optimal one. For larger scale instances, we consider other practically oriented heuristics. Our numerical experiments show that our proposed algorithm performs very well compared to state-of-the-art methods in the literature.

Keywords: multi-location inventory, proactive transshipments, lost sales, dynamic programming, concave piecewise-linear approximation

[*]Corresponding author.

1 Introduction

The dominant cause of mismatch between customer demand and a retailer's available stock is demand uncertainty. While modern information technologies offer possibilities to store big data and forecast the average demand rate quite accurately for groups of products, there are still markets in which demand uncertainty results in unsold inventories and loss of sales, reducing retailer profits. Companies usually use historical data for recent years and allocate their inventories across multiple regions based on the previous year's sales patterns. However, random fluctuation in customer demand, such as that caused by weather conditions and other external factors, might produce significant deviations from the pattern, which as was empathized by CSNews (2001), can lead to both overstocks and stockouts depending on the regions.

These days, the news media report many stories of shortages of various products—the lack of winter boots in New York during a cold snap, as described by Pastor (2014), or a shortage of umbrellas at a golf championship in Louisville during a rainy season, as highlighted by Democrat & Chronicle (2013). A representative example of a weather-related product is an air conditioning system. Sales of air conditioners are increasing consistently throughout the world, as customers today can increasingly afford to purchase them. Especially during the summer, demand for cooling systems increases with the temperatures and often exceeds the supply. Although uncertainty about the demand is high and frequently leads to lost sales, in some regions, demand is quite stable and can be accurately forecasted, for example, in Dallas, where most buildings and homes have central cooling systems. Thus, because of seasonal demand fluctuations, companies may observe shortages in certain areas and still have excesses in others. A shortage is usually covered by an additional emergency shipment from the supplier, however, suppliers can also face these shortages. For example, Mei-hsing (2011) reported a large shortage of air conditioners in China that could not be covered because of the lack of key components supplied by upstream manufactures. That shortage in turn causes the inability to satisfy customer demands and, as a result, lost sales for a company.

Similar losses happen in the pharmaceutical industry. According to US Food and Drug Administration (2014), the number of drug shortages quadrupled from 61 in 2005 to 250 in 2011; although that number decreased to 117 in 2012, it remained a significant issue for the US public health system. At the same time, Healthcare Distribution Management Association (2009) reports that the pharmaceutical industry experiences approximately $2 billion in drug returns annually due to their expiration dates.

To alleviate the mismatch between an actual customer demand and an available stock in multiple locations under the inability to replenish from a central warehouse, we propose lateral transshipments, e.g., the movement of stock between locations of the same echelon. So far, the use of lateral transshipments has been observed in the management of spare parts. As slow-moving expensive items, spare parts require a quick response to an infrequent demand. Lateral transshipments allow for an additional source of procurement from a nearby location once a stockout of spare parts occurs, that is, *reactive transshipments.* This leads to cost reductions and service improvements compared to no-transshipment networks. Until recently, the main limitation in applying this strategy for final products and consumable goods has been insufficient and unreliable information about current inventories within supply chains. Thus, ongoing developments in information technology such as the widespread introduction of ERP systems provide opportunities to exchange real-time data about inventories. This information might be used by lateral transshipments not only to satisfy the demand of customers who are willing to wait but also to predict stockouts, that is, *proactive transshipments.*

In our paper, we focus on items with a short selling seasons and highly uncertain demand. We refer to the following example use of lateral transshipments. Consider a multi-location distribution network with one central warehouse in Bavaria, Germany and 15 retail outlets in different German states (Fig. 1). The central warehouse orders replenishments from a supplier based on demand forecasts over a month for retailers. The order is placed far in advance, incorporating a long lead time for production and delivery. Once the order arrives at the central warehouse, it makes decisions about the amount of goods that should be delivered to retailers (Fig. 1a). Assume that at the beginning of the month all retailers receive the product. Each day of the month, customers come to retailers and run down their stocks. Depending on customer's needs for the product, the depletion of stock happens faster in some states than in others. To avoid stockout situations that result in the loss of customers and related sales, retailers can share their inventories in the beginning of each day: if the product amount at one retailer is insufficient to meet the demands of customers over the upcoming days, the product can be transferred from other outlets in the distribution network (Fig. 1b).

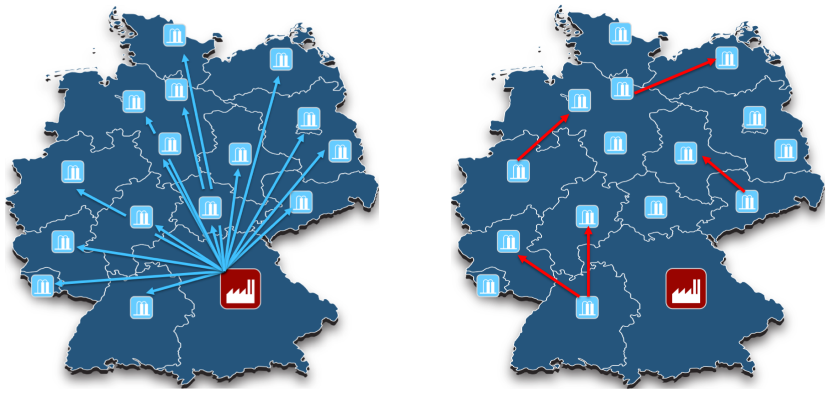

(a) Distribution from central warehouse. **(b)** Lateral transshipments.

Figure 1: An example of a multi-location distribution network.

In our work, we analyze this particular situation and develop a model for a transshipment problem, which can be classified as a multi-location, multi-period transshipment problem with lost sales in cases of stockouts. We present a dynamic programming approach to finding an optimal transshipment policy. In each period, the policy decides whether a transshipment should be performed, between which locations and how many units should be shipped to maximize the profit of the whole network over all periods. As the size of the state space and decision space is large for real-size problems and, therefore, causes a problem known as the *curse of dimensionality* for the dynamic program, we derive a near-optimal transshipment policy by applying a forward approximate dynamic approach. Numerical experiments show that our proposed algorithm is significantly close to the optimal one and achieves higher profits compared to known practices.

To the best of our knowledge, this important problem, as we have described it, has not yet been addressed in the literature.

2 Literature review

In contrast to the proactive transshipment model formulated herein, the majority of transshipment models proposed in the literature consider reactive transshipments. These have been studied under both periodic and continuous replenishment settings. Krishnan and Rao (1965) were the first to publish a paper on reactive transshipments under periodic review. They derive the formula for optimal stock levels of multiple locations with a transshipment at the end of a single replenishment period, once the demand is known. As transshipment costs are assumed to be the same between all locations, it is enough for the transshipment policy to choose one location with a surplus to satisfy the shortage in another location. This research was extended by Robinson (1990) and Herer et al. (2006). Robinson (1990) models a multi-location problem with non-identical locations in terms of the cost and demand distributions and multiple periods. The authors prove the optimality of the base stock (order-up-to level) policy and demonstrate the stationarity of the optimal order-up-to point. Analytical solutions can be characterized for special cases of two locations and multiple identical locations. Due to the complexity of the problem in more general multi-location settings, authors suggest an approximate solution that uses Monte Carlo sampling to avoid the time consuming calculation of the expectation. However, the accuracy of this approach depends on the demand sample, and there is no guarantee of convergence to the optimal solution. Herer et al. (2006) suggest a simulation-based approach that gives values of the order-up-to level that converge to the optimal solution. Once the order-up-to level is determined, the transshipment problem is solved as a deterministic linear-cost network flow problem. Their approach to solve the transshipment problem is similar to us, however, aside from the proactive nature of transshipments that happen before the demand is known, our transhipment problem differs from the above as it considers the expected profit of next periods, while Herer et al. (2006) run transhipment to minimise a current period cost.

While previous efforts had considered the transshipment as a single time event per order cycle, Archibald et al. (1997) formulates a model of a period review where each demand triggers a transshipment decision if a stockout occurs. The work has been extended by Archibald (2007) and Archibald et al. (2009) for multiple locations. There are few other studies of reactive transshipments that consider multiple transshipments per cycle, e.g. Grahovac and Chakravarty (2001), Çömez et al. (2012), Glazebrook et al. (2014). Our work differs in that it allows for transshipment decisions every time period at predetermined points, before the demand is known. Moreover, we consider simultaneous transshipments between locations, in a way that one location can get shipments from multiple other locations. Due to the complexity of such setting, it has not been considered in the above discussed works.

Reactive transshipments under continuous order review settings have been studied by Lee (1987), Axsäter (2003), Minner et al. (2003), Wong et al. (2005), Wong et al. (2006) and others. For the most part, these models with reactive transshipments are motivated by spare parts applications. Recently, Paterson et al. (2012) introduces a mixture of reactive and proactive transshipments under continuous review replenishment policy. This hybrid transshipment policy is triggered by a stockout event, as in the case of reactive transshipments, but it might ship more than required by unsatisfied demand, thereby imitating the preventive behavior of proactive transshipments. Seidscher and Minner (2013) introduce a proactive transshipment policy in a continuous time review setting that is based on redirection of customer demand. While the proposed transshipment policy is proactive in that it is triggered by each demand event and not in a response to stockout, the timing of transshipment is not an issue here as it is for a reactive transshipment policy.

To the best of our knowledge, all other studies that consider proactive transshipments address periodic review replenishment settings that make the decision on the timing of network stock rebalancing easier to implement: in most cases, transshipments occur either at the beginning or at the end of the replenishment period. An exception is the policy suggested by Agrawal et al. (2004), which looks for an optimal time between replenishments to rebalance the stock. The authors show that for multiple identical locations, rebalancing tends to take place close to the end of the order cycle. In our work, the transshipment can happen more than once within the order cycle, and the time of each transshipment is fixed, however, the transshipment policy decides by itself whether it is optimal to run transshipments now or to postpone to the next transshipment point.

Besides the timing of transshipments, the decision should incorporate the source and the amount of inventory to transship. All together, this makes an inventory problem with proactive lateral transshipments difficult to solve to optimality. Therefore, studies have mostly focused on either two-location networks or single-period problems or both, where it is attainable to find and analyze an optimal policy. One of the first works about proactive transshipments conducted by Allen (1958) finds an optimal transshipment policy for a single period mutli-location problem. Later, Gross (1963) includes the replenishment decision, and looks for optimal order-up-to points and an optimal transshipment policy. The analytical solution is provided for the two-location case; for multiple locations, the authors suggest an iterative procedure that calculates partial derivatives and adjusts the variable with the highest marginal value at each iteration. For multiple locations the search of the source location is usually simplified by the assumption of identical costs. One of the first papers that consider nonidentical costs of locations is by Bertrand and Bookbinder (1998). As opposed to our approach, they allow a transshipment decision once, at the end of the order cycle, that results in a single period transshipment problem. For a multi-period, multi-location problem with backorders, Karmarkar (1980) shows that the problem at each period is similar to the one-period problem. Therefore, the form of the optimal policy is the same for muti-period and one-period problems. Our work differs from the above as it considers a multi-period, multi-location problem with lost sales, and, since it can not be simplified to the set of single period problems, the problem is solved using the methodology of dynamic programming.

Due to the complexity of the problem, there are some heuristic transshipment policies as well. Banerjee et al. (2003) introduce two proactive transshipment rules, namely Transshipment Based on Availability (TBA) and Transshipments for Inventory Equalization (TIE). Under these policies a transshipment is triggered when the inventory level of one of the locations falls below the average demand rate. The TBA policy behaves in a reactive way by redistributing stock between locations to prevent the next period's stockouts, and has a short time effect, whereas the TIE policy balances inventories to reduce the severity of stockouts over the long run. These policies are easy to implement and can be applied to networks with multiple locations and multiple periods. At the same time, there are no assumptions concerning the specific value of the cost parameters related to the various stock locations. Moreover, the cost is not directly considered in the implementation of these policies. The analysis of the cost performance of the policies can be conducted only as a result of operating the policies, see Burton and Banerjee (2005).

Recently, new proactive and reactive transshipment policies are obtained by Van der Heide and Roodbergen (2013) for a multi-location library control systems with fixed stock and return. The optimal policies are computed for small-sized networks by applying a stochastic dynamic programming algorithm. For real-life sized network configurations, two heuristics are presented. One of them considers clustering of locations, another examines a couple of periods ahead to define whether it is beneficial to transship and rebalance now.

For an overview of the existing literature on lateral transshipments we refer to the comprehensive review by Paterson et al. (2011).

To solve the the multi-period dynamic problem, we apply dynamic programming technique introduced by Bellman (1957). Modelling the problem as a Markov decision process and then solving it using linear programming, policy iteration or value iteration is common in the lateral transshipment literature. It has been employed before by e.g. Archibald et al. (2010), Paterson et al. (2012), Van der Heide and Roodbergen (2013), Seidscher and Minner (2013). However, the inability of solving large-scale problems discussed by e.g. Seidscher and Minner (2013) results in the development of heuristic methods. For example, Agrawal et al. (2004) use a Monte Carlo simulation-based approach to find a dynamic rebalancing policy. The expected lost sales cost is approximated by the sample average function corresponding to a generated random sample. The policy demonstrates significant advantages over a static rebalancing policy. Paterson et al. (2012) propose a quasi-myopic heuristic that, when the stockout occurs, compares the benefit of making a transshipment now and never again with performing a no-transshipment.Archibald et al. (2010) implement an approximate dynamic programming (ADP) approach that applies a dynamic programming policy improvement step to a static policy. Using a numerical study, the authors show that the transshipment policy derived by ADP approach is close to the optimal policy for small number of locations, and in most cases, it outperforms other heuristics for real-life sized problems. Since the approximation methods are based on the known properties of optimal policies for specific cases, the obtained policies often perform better than simple heuristics. In our work, we look at the methods of approximate dynamic programming discussed by Powell (2011). The applied approximation technique has not been used for any transshipment problem yet.

With respect to the literature, the contributions of our paper can be summarized as follows: (1) We analyze a multi-location network with multiple opportunities for proactive transshipments within the order cycle. The transshipment point is set at the beginning of each period, before the realization of demand is known. The transshipment is implemented only if it is profitable for the whole network, otherwise it is postponed. (2) We consider simultaneous transshipments between multiple locations. (3) Our policy works with non-identical locations, that differ in terms of cost and demand distribution. The transshipment cost is different for each couple of locations and depends on the distance between locations. (4) Using the methods of approximate dynamic programming, we develop a fast and efficient algorithm to solve industrial-size multi-location transshipment problem. The chosen approximation simplifies the search of each period transshipment decisions to the solution of a network flow problem.

The paper is organized as follows. In Section 3, we present the multi-location transshipment problem and assumptions. We introduce notation and formalize the problem in terms of dynamic programming in Section 4. Section 5 discusses limitations of the optimal policy and presents practically oriented heuristics applied for the transshipment problem. In Section 6, we use the methods of approximate dynamic programming to develop a dynamic proactive transshipment policy. In Section 7, we design an extensive numerical study and discuss the results that report the performance of a dynamic proactive transshipment policy in comparison with other known policies. Finally, Section 8 concludes the paper.

3 Problem formulation

We consider a centralized distribution network, where all retail locations are operated by a central entity whose goal is the maximization of the overall profit of the network.

Time is discretized and divided into order cycles. Each order cycle contains a finite number of periods. At the beginning of each period, transshipments are allowed. We assume that the replenishment decision is made at the beginning of the order cycle and that no further replenishment takes place before the next order cycle. Within each period, events occur in the following order:

1. Observe the inventory level at each retailer.
2. Determine and make the transshipment decision, if required. The corresponding transshipment cost is paid.
3. Collect the demand of customers for the product at each location.
4. Satisfy the aggregated demand.
5. Obtain a reward for meeting the demand. All unsatisfied units of demand are lost.
6. The holding cost is paid if there is an excess inventory that needs to be carried to the next period.
7. End with the stock after demand satisfaction.

We make these other assumptions to model the problem as described above:

- The model describes the behavior of the system for one product that is identical at all retail locations.
- No backorders are allowed. If the retailer cannot satisfy demand immediately from the stock on hand, then the demand is lost (a lost-sales case). If customer demand is greater than the inventory level of the retailer, then the retailer uses all stock on hand to satisfy the demand and faces a stockout at the end of the day. In this case, we consider that this retailer does not admit any customer demand until the fulfillment of the stock from other retailers during the month or from the central warehouse at the beginning of the next month.

- The time for replenishment orders from the central warehouse to retailers and for transshipment orders between retailers is negligible (or less than the retailer closing time). This assumption simplifies the mathematical formulation of the problem; therefore, it is used in the majority of the literature on lateral transshipment. The analysis of the system with multi-period in-transit times is a challenge for future research.
- The transshipment cost depends on the distance. As we are completing transshipments over a geographically large network within a short time, we should take into account how costly it is to ship the item from one retailer to another. This cost usually depends on the type of transportation that is used for transshipments, which in turn depends on the distance between the locations.
- The distribution of the demand at each time period for each location is known. Even if there is no exact information about future demand, companies are often aware of the behavior of the demand during previous time periods. This information can be used to estimate probability distributions.
- The demand is independent between locations, as locations are situated in different regions and, therefore, have different sales rates.
- We consider most final products and consumable goods separable, discrete items, such as books, tires, coolers. Formally, this means that we use a discrete variable to describe the quantity of the product.
- We examine one order cycle that is divided over a finite number of periods. Once the optimal decision is found for one order cycle of the selling season, it can be sequentially implemented for the whole season.
- The replenishment policy is to determine an order-up-to level for each location at each order cycle. In this paper, we focus on the transshipment problem and consider the order-up-to level given for each retailer. The de facto order-up-to level for each location can be found post hoc by simulation over the range of possible values for the inventory level, which might be limited by, e.g., the capacity of a location. This is in line with the recent literature about lateral transshipment, e.g., Archibald et al. (2010) and Glazebrook et al. (2014). Starting from the investigation of the optimal replenishment policy in the presence of transshipment, the research now tends to skew toward the study of transshipment policy, the difficulty of which increases rapidly with the size of a network.

The objective of this study is to identify the transshipment policy that maximizes the expected profit of the system over a finite number of periods and performs well for an arbitrary order-up-to-level vector.

Based on the problem formulation, we have a dynamic, finite horizon problem that involves making sequential decisions over time. The main methodology that considers the sequential decision behavior of the dynamic system is the dynamic programming. Therefore, in the next Section, we use the terminology of dynamic programming to model the problem presented above. The limitations and extensions of the dynamic programming approach to find an optimal solution will be discussed later.

4 Model

We now introduce the mathematical model to describe the transshipment mechanism. For the vocabulary and notation that we are using to formalize the problem and develop the policy, we refer to Powell (2011).

To describe the evolution of the system we use the following notation:

Constants:

L	number of retailers
T	number of periods
$\mathcal{L}$	set of locations, $\mathcal{L} = \{1, \ldots, L\}$
$\mathcal{T}$	set of time periods, $\mathcal{T} = \{0, \ldots, T\}$

Let $t \in \mathcal{T}$ be a time index indicating the period of a transshipment decision and $i, j \in \mathcal{L}$ be locations indexes corresponding to the retailer number. The terms locations and retailers are used interchangeably throughout the paper. Further, we use bold characters to represent vectors.

Parameters:

S_i	initial order-up-to level of the inventory of the location i, $S_i \geq 0$
$\mathbf{S}$	initial state of the system at time $t = 0$, $\mathbf{S} = (S_1, \ldots, S_L)$
h_i	holding cost per unit in stock at location i
p_i	sales price per unit in location i
c	cost of transshipment per unit per distance
ρ_{ij}	distance between location i and location j, $\rho_{ij} > 0$

As our model is a profit maximization problem, we do not need to consider penalty costs for stockouts explicitly, but they are captured implicitly by the loss of profit.

Pre-decision state variables:

x_{it}	inventory in hand at location i at the beginning of the period t
$\mathbf{x}_t$	state of the system at time t, $\mathbf{x}_t = (x_{1t}, \ldots, x_{Lt})$

The state at time t represents the inventory level before any transshipments are made. It is so-called pre-decision state (Powell (2011)), which is usually used in dynamic programming to describe the state space. The inventory level at each location and in each time period is a non-negative integer value. The state at the beginning of the first period $\mathbf{x}_0$ is equal to the initial order-up-to level $\mathbf{S}$.

Post-decision state variables:

We define a post-decision state variable herein as inventory on hand in location i at period t after making a transshipment decision but before observing the realization of the demand:

y_{it}	inventory on hand in location i after transshipments at time t
$\mathbf{y}_t$	post-decision state of the system at time t, $\mathbf{y}_t = (y_{1t}, \ldots, y_{Lt})$

Decision variables:

z_{ijt}	transshipment from location i to location j at the beginning of period t
$\mathbf{z}_{it}$	vector of transshipments from location i at time t, $\mathbf{z}_{it} = (z_{i1t}, \ldots, z_{iLt})$
$\mathbf{z}_t$	transshipment decision in the system at time t, $\mathbf{z}_t = (\mathbf{z}_{1t}, \ldots, \mathbf{z}_{Lt})$

For each location i, the transshipment decision at time t, $\mathbf{z}_{it}$, shows how much and to which locations current inventories should be shipped. Transshipment to the same location z_{iit} indicates the inventory kept by this location to satisfy its own upcoming demand. There are the following constraints on the transshipment decision: The transshipment amount z_{ijt} can take only non-negative integer values. The total amount shipped from the location i at time t, $\sum_{j\in\mathcal{L}} z_{ijt}$, is equal to the current inventory level at this location x_{it}.

Exogenous process:

d_{it}	realized demand in location i at period t
$\mathbf{d}_t$	vector of demands in the system at time t, $\mathbf{d}_t = (d_{1t}, \dots, d_{Lt})$

As we consider that inventory level of final products and consumable goods can take only integer values, we use a discrete distribution to describe the stochastic process. Demand is assumed to be non-negative, and to be independent between retailers and over time. The last assumption is made for computational reasons.

Transition function:

The transition function describes how the state of the system changes with each decision. Using the pre-decision and the post-decision state definition, we express the system dynamics in two steps:

$$y_{it} = x_{it} + \sum_{j\in\mathcal{L}} (z_{jit} - z_{ijt}), \tag{1}$$

$$x_{it+1} = (y_{it} - d_{it})^+, \ \forall\, i \in \mathcal{L}, \tag{2}$$

where x^+ denotes a positive part and defined by the formula:

$$x^+ = \max(x, 0) = \begin{cases} x & \text{if } x > 0, \\ 0 & \text{otherwise.} \end{cases}$$

The first function captures the pure effect of transshipment decisions, while the second function captures the effect of demand satisfaction.
Further, we use the following shortcut notation for the transition function:

$$\mathbf{y}_t = X^{M,z}(\mathbf{x}_t, \mathbf{z}_t),$$
$$\mathbf{x}_{t+1} = X^{M,d}(\mathbf{y}_t, \mathbf{d}_t).$$

Reward and cost functions:

The cost of transshipments at time t:

$$C_t(\mathbf{x}_t, \mathbf{z}_t) = \sum_{i,j\in\mathcal{L}} c\,\rho_{ij}\, z_{ijt}\,, \ \text{ where}$$

$$\sum_{j\in\mathcal{L}} z_{ijt} = x_{it}\,, \ \forall\, i \in \mathcal{L}, \tag{3}$$

$$z_{ijt} \in \mathbb{Z}_{\geq 0}\,, \ \forall\, i, j \in \mathcal{L}.$$

The cost occurring at the beginning of the period is the cost of transshipment of an item from one retailer to another. It depends on the distance between locations. The transshipment amount between two locations is subject to the current inventory level at the source location.

The reward during the time interval $[t, t+1)$:

$$\begin{aligned} R_t(\mathbf{y}_t, \mathbf{d}_t) = \sum_{i\in\mathcal{L}} p_i \min\Big(y_{it},\ d_{it}\Big) & - \\ - \sum_{i\in\mathcal{L}} h_i \Big(y_{it} - d_{it}\Big)^+. & \end{aligned} \tag{4}$$

We assume that customers pay directly after the satisfaction of their demands. The reward of the system after selling items equals the sales of the system minus the holding cost for each item in stock at each retailer.

Profit function:

In terms of the above notation, the total network profit at any period t for a particular realization of demand can be stated as follows:

$$P_t(\mathbf{x}_t, \mathbf{d}_t, \mathbf{z}_t) = R_t(\mathbf{y}_t, \mathbf{d}_t) - C_t(\mathbf{x}_t, \mathbf{z}_t). \tag{5}$$

Objective function:

Let Π be the set of policies and let $\mathbf{z}_t^\pi$ be a particular decision rule at time t indexed by π. The objective of the transshipment problem is to find a policy π from the set Π that will maximize the expected sum of profits over all time periods:

$$\max_{\pi\in\Pi} \mathbb{E}\Big\{\sum_{t=0}^{T-1} P_t(\mathbf{x}_t, \mathbf{d}_t, \mathbf{z}_t^\pi)\Big\}. \tag{6}$$

Once we find the optimal policy, we can use it to determine the most profitable decision at any time period in any state at which we can end up.

The problem (6) is a stochastic optimization problem, which has a non-linear structure and accepts integer variables only. Using a standard dynamic programming formulation, we can solve it recursively by computing the optimality equations (7):

$$V_t(\mathbf{x}_t) = \max_{\mathbf{z}_t} \Big(- C_t(\mathbf{x}_t, \mathbf{z}_t) + V_t^z(\mathbf{y}_t)\Big), \ \text{ where} \tag{7a}$$

$$V_t^z(\mathbf{y}_t) = \mathbb{E}_{\mathbf{d}_t}\Big\{R_t(\mathbf{y}_t, \mathbf{d}_t) + V_{t+1}(\mathbf{x}_{t+1})\Big\}, \ \forall\ 0 \le t \le T-1, \tag{7b}$$

where $V_t(\mathbf{x}_t)$ is known as a value function that indicates the value of being in a pre-decision state $\mathbf{x}_t$. Here, $\mathbf{y}_t = X^{M,z}(\mathbf{x}_t, \mathbf{z}_t)$ describes the dependency of the post-decision state $\mathbf{y}_t$ on the pre-decision state $\mathbf{x}_t$ and the decision $\mathbf{z}_t$, and $\mathbf{x}_{t+1} = X^{M,d}(\mathbf{y}_t, \mathbf{d}_t)$ describes the dependency of the pre-decision state $\mathbf{x}_{t+1}$ on the post-decision state $\mathbf{y}_t$ and the demand $\mathbf{d}_t$. The value function at time period T is equal to zero for any inventory level.

We name $V_t^z(\mathbf{y}_t)$ as a post-decision value function. Note that optimality equations can be written around the pre-decision state (classical representation) as well as around the post-decision state. We refer to Powell (2011) for further discussions. However, the formulation of the problem around the post-decision state allows a deterministic maximization problem for a known demand realization, which can be calculated by known methods of mathematical programming. Therefore, we decide on the optimality equations around the post-decision state, and later, when we talk about the approximation, we mean the approximation of the post-decision value function.

5 Transshipment policies

Due to the well known curse of dimensionality, the solution of the optimality equations (7) can be obtained only for small-sized networks. Therefore, many heuristic transshipment policies have been developed to solve the problem for real-world sized networks close to optimality within a reasonable time frame. Here, we introduce three of them that we use further as comparison with the optimal and near-optimal policies in Section 7.

The first heuristic is a a reactive transshipment policy, which transships from the closest location with stock on hand once in the beginning of the period to the location that has no inventories. Before completing transshipments, the source location should define the amount that will be shipped to the location in need. Usually, this policy is applied for problems with allowed backorders, see e.g. Lee (1987), Seidscher and Minner (2013), where the source location decides whether to completely satisfy the destination's backordered demand or to keep part of the stock to satisfy its own demand in the subsequent periods. In our case, the decision occurs proactively before we know the realization of demand. Therefore, we set the amount to be transshipped as a parameter that is positive and less than the available stock of the source location. The parameter can be fixed for each run of the policy or vary from period to period. We name this policy as a Closest location transshipment policy.

The second heuristic is one of few proactive heuristics for multiple locations presented in the literature. Lateral TIE policy has been proposed by Banerjee et al. (2003). Once the inventory of one location falls bellow the expected daily demand rate, the redistribution of inventories is triggered in order to equalize the runout time among all locations. The authors assume that redistribution occurs through a complete tour by a single transport vehicle, where the whole available stock is first picked up from source locations. Then, the required amount is delivered to destination locations. As we consider a large network of locations, we drop this assumption in our implementation and consider that the redistribution is completed by a number of transshipments between locations. The main limitation of TIE policy discussed by Tiacci and Saetta (2011) is that the cost has not been directly considered in the decision rule, and it can be evaluated only post factum (Burton and Banerjee (2005)). Therefore, we modify the policy by including transshipment costs in the decision. To define the sequence of transshipments between locations, we rank locations with the required amount to be shipped for the equalization in descending order. Then, for the location with the greatest need, we rank the locations with available stock in ascending order in terms of the transshipment cost and transship the required quantity based on the ranking, if possible. Thus, we first complete transshipments between the closest locations and then look at farther locations if more items are required. In early experiments, we found that the modified TIE outperforms the TIE presented by Banerjee et al. (2003) for our model. Therefore, we include the modified TIE in our further comparison.

The third heuristic is a Lookahead transshipment policy. Lookahead policies have been applied before for both reactive and proactive transshipments, see e.g. Axsäter (2003), Van der Heide and Roodbergen (2013). Here, the policy looks at the expected profit of making a transshipment of one unit between two locations in the beginning of each period and compares the obtained value with a no-transshipment expected profit. If the difference between expected profits is higher than the transshipment cost, then the amount that should be transshipped between these two locations is increased by one unit. The procedure is repeated until we can not find locations with these properties. Finally, we aggregate and complete the transshipments between locations. The expected profit for each period is calculated as a one-period expected profit for meeting aggregated demand from the current period until the end of the time horizon under the assumption that there are no future transshipments.

6 Approximate Dynamic Programming

Approximate dynamic programming (ADP) is a methodology for solving large stochastic optimization problems, which seeks to address the *curse of dimensionality* of dynamic programming by using an approximation of the value function instead of the true value function. Different forms of ADP have been studied in control theory by, e.g., Bertsekas (2012); in artificial intelligence, by e.g., Sutton and Barto (1998); and in operations research, by e.g., Powell (2011). For the vocabulary and notation that we are using, we refer to the last section.

In this paper, we apply forward ADP. It is based on the idea of going forward in time instead of stepping backward and looking through all states as we do in a DP algorithm. To simulate the process forward in time, we need to generate a certain number of demand samples. The number should be large enough to learn the value function of being in a particular state and small enough to process the algorithm in a reasonable time. Starting with the given approximation of the value function, the algorithm simulates the sample path of demand realizations for next T periods. To generate samples from a known distribution of the demand, we use Monte Carlo simulation. Further, for each period, the sampled demand realization is used to calculate the next state, except period $t = 0$, when the starting inventory level is initialized by the stated order-up-to level. Then, a greedy control policy selects the best decision by solving the maximization problem with the estimate of the value function of the previous iteration instead of the true value function. To solve the maximization problem, the expectation inside the maximization operator should be computed. For our problem, we assume that the distribution of the demand is known for each location and each period. If the random state space is small enough, the expectation can be computed exactly or approximated using the sample of possible demand realizations. In other cases, the problem might be reformulated in terms of the post-decision state introduced by Powell (2011). The estimation computed by the maximization problem is further used to update the current estimate of the value function. If the expectation is approximated, then exponential smoothing is applied for the update. Using the generated demand sample, the next state is computed by the transition function. After N iterations, the algorithm returns the best estimate of the value function. The forward ADP algorithm with the use of a post-decision state is presented in Appendix A.

There are two parameters in the general forward ADP algorithm that should be specified before going further. The first, ϵ, is used to explore possible states that otherwise might never be enumerated by the ADP procedure. Following the sample path and making only the "greedy" decision, we choose the most profitable states according to the current estimate of the value function. However, this strategy may lead to a local solution because of a poor estimate of the value function, and better states might never be examined. To avoid such situation and to explore other possible states, we apply the ϵ-greedy method: behave greedily most of the time, and choose the best action that maximizes the expected profit. However, every once in a while, with the small probability ϵ, select the random action. This gives us an estimate of the value of being in states that we would otherwise be unlikely to visit. Experimentally, we choose the ϵ value equal to the declining exploration probability:

$$\epsilon_n = \frac{1}{b^n},$$

where n is an iteration number and b, $b \in (0, 1)$.

The second parameter is the stepsize, α, which helps uncover the value function through iterations and smooth the error that we introduce by sampling the demand. For a review of possible stepsize rules that have been applied so far, we refer to George and Powell (2006). Although there is no optimal stepsize rule, Powell (2011) suggests a range of simple stepsize formulas that yield good results for many practical problems. In this paper, we apply one of these rules—a generalized harmonic stepsize:

$$\alpha_n = \frac{a}{a+n-1},$$

where n is an iteration number and a, $a \in \mathbf{R}^+$ controls the convergence rate.

In our experimental section, b is fixed to 0.7, and a is fixed to 5.

6.1 Approximation based on properties of the value function

The size of the action space of the transshipment problem forces us to look for an approximation of the value function that, first, considers the properties of the true value function and, second, makes the maximization problem on each iteration solvable within a reasonable time.

We start by learning the properties of the true value function. Let us assume that there are no transshipments between locations. Then, the value function degenerates into the sum of the expected profits in each location at each time period. Note that each member of this sum is a concave function of the inventory level. Therefore, the value function subject to no-transshipments is a concave, separable function on the state space, where the inventory level in each location can take only integer values.

Introduction of transshipment decisions into the system does not change the concavity. This statement is a result of the concavity of the maximization problem at each time period, which in turn, follows from a property of a concave function under perturbation (see, e.g., Rockafeller (1996)), assuming that the value function at each time period is concave. However, the concavity of the value function at each time period can be ruined by the non-linear transaction function that we have in a lost-sales case, as it is a case in general for a composition of concave and convex functions, where the first one is not a monotonically non-increasing function. Our extensive numerical studies show that the value function preserves its concavity property. We believe that this can be proven based on the results of Zipkin (2008).

To imitate the properties of the value functions we have observed in many trials, we apply a piecewise-linear concave approximation, which is often mentioned as a reasonable choice in Powell (2011, pp. 501). Furthermore, even though the value function is non-separable, the separability of the value function in the no-transshipment case suggests the use of a separable approximation of the value function, which is computationally easy to address. Therefore, for each time period t, we approximate the value function by the sum of piecewise-linear, concave functions:

$$\hat{V}_t^z(\mathbf{y}_t) = \sum_{i=1}^{L} \hat{V}_{it}^z(y_{it}),$$

where the domain of each function $\hat{V}_{it}^z(y_{it})$ is a set of non-negative real numbers with breakpoints in integers.

6.1.1 Network flow problem

Let us assume that at time t we know the value of function $\hat{V}_{it}^z(\cdot)$ for each location i at $K_{it} = \{0, \ldots, k_{max}\}$ points. Then, the function $\hat{V}_{it}^z(\cdot)$ at each time can be defined by the set of breakpoints (v_{it}^k, u_{it}^k), where $k \in K_{it}$ are indexes of breakpoints; $u_{it}^k \in \{\mathbf{Z}^+ : u_{it}^0 = 0, u_{it}^{k+1} > u_{it}^k\}$ are the integer points of the domain. $v_{it}^k = \hat{V}_{it}^z(u_{it}^{k+1}) - \hat{V}_{it}^z(u_{it}^k)$ are the slopes of the value function that arranged in the decrescent order $v_{it}^0 > v_{it}^1 > \cdots > v_{it}^{k_{max}}$ to insure the concavity. Note that the domain of $\hat{V}_{it}^z(\cdot)$ is bounded by the aggregated inventories of the network.

After we choose the approximation of the value function, we turn back to the maximization problem. The approximation of the value function by the sum of separable, piecewise-linear, concave functions allow us to reformulate the maximization problem as a minimum-cost flow problem, which significantly simplifies further calculations of solutions.

Using the notation for the problem in terms of the post-decision state and introducing additional variables $g_{it}^k \in [0, u_{it}^{k+1} - u_{it}^k]$, $k \in K_{it}$, the maximization problem for each time t from the optimality equations (**??**) can be stated as:

$$\max_{z_{ijt}} \left(-\sum_{i\in\mathcal{L}} \sum_{j\in\mathcal{L}} c\rho_{ij} z_{ijt} + \sum_{i\in\mathcal{L}} \sum_{k\in K_{it}} v_{it}^k g_{it}^k \right), \tag{8}$$

subject to

$$\sum_{j\in\mathcal{L}} z_{ijt} = (y_{it-1} - d_{it-1})^+ \quad \forall i \in \mathcal{L}, \tag{9}$$

$$\sum_{k\in K_{it}} g_{it}^k = \sum_{j\in\mathcal{L}} z_{jit} \quad \forall i \in \mathcal{L}, \tag{10}$$

$$0 \le g_{it}^0 \le u_{it}^1 \quad \forall i \in \mathcal{L}, \tag{11}$$

$$0 \le g_{it}^1 \le u_{it}^2 - u_{it}^1 \quad \forall i \in \mathcal{L}, \tag{12}$$

$$\vdots$$

$$0 \le g_{it}^{k_{max}-1} \le u_{it}^{k_{max}} - u_{it}^{k_{max}-1} \quad \forall i \in \mathcal{L}, \tag{13}$$

$$g_{it}^{k_{max}} \ge 0 \quad \forall i \in \mathcal{L}, \tag{14}$$

$$z_{ijt} \in \mathbb{Z}_{\ge 0} \quad \forall i, j \in \mathcal{L}. \tag{15}$$

The constraints (9) are the flow-balance constraints for locations at time t before transshipment. They state that the cumulative amount that is shipped from location i should be equal to the leftover inventory of location i after demand satisfaction. The constraints (10) are the flow-balance constraints for locations in time period t after transshipments. They indicate that all inventories transshipped to location i at time t from other locations build the inventory level of location i to satisfy next up-coming demand. The inventory level at each location i at time t after transshipments is used to calculate the value function, which is represented by the arcs g_{it}^k. From the approximation, these arcs have cost v_{it}^k, and the upper bound is covered by constraints (11–13). Note that the last arc for each location i has an infinite upper bound to make the problem feasible (14). Finally, the constraints (15) are non-negative, integer constraints on transshipment flows between locations. However, due to the integrality property of the minimum-cost flow problem (8), these constraints are unnecessary and can be relaxed. The network problem (8) is presented in Fig. 2.

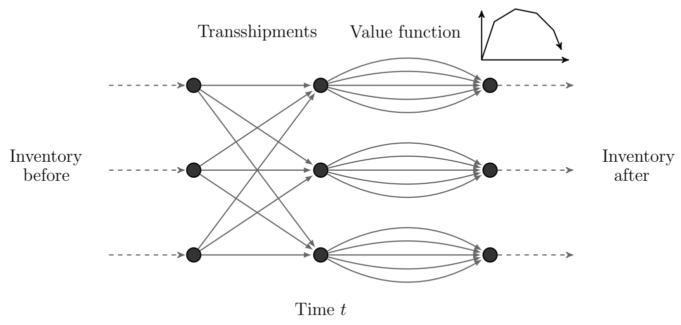


Figure 2: Network representation of the transshipment problem, where the nodes of one column are the locations at different points of period t, and the arrows indicate the flow of inventory.

By applying separable, piecewise-linear approximation, we reformulate the problem and recognize the network structure. As the sub-problem (8) has a linear objective function and linear constraints, it is a linear programming problem. Therefore, any linear solver can find optimal solutions in a reasonable time. Moreover, the solutions are naturally integers without applying any integer programming techniques.

6.1.2 CAVE update of the approximation

The next step in the forward dynamic algorithm is an update of the approximation. The rate of learning for the value function approximation depends on the way we update the approximation. That, in turn, leads to better and faster convergence of the approximation to the true value function. Moreover, as the approximation has a concavity property, the updating algorithm needs to take this into account. For the transshipment problem, we choose the Concave Adaptive Value Estimation (CAVE) algorithm that updates the concave, separable, piecewise linear function using the stochastic gradient estimates of this function at different states of the state space. The algorithm has been proposed by Godfrey and Powell (2001) for the newsvendor problem and for the two-stage stochastic distribution program with the network recourse. Later, Godfrey and Powell (2002b) and Godfrey and Powell (2002a) consider the CAVE update to a multistage distribution problem with an application to fleet management. They show that the approximations using the CAVE update yield high-quality solutions compared to the deterministic and stochastic rolling-horizon methods.

The idea of the CAVE update is to use the sample gradient information to learn the value function. In the case of concave, piecewise-linear approximation, which is represented by the sequence of slopes, it is much easier to update the slope of the value function approximation in each iteration than the value function itself, as is done in the general ADP algorithm. The only things that are required for the update are to define the update intervals and to find the sample gradients. The second issue will be discussed later. First, we introduce the notation for gradients that is used for the update. As the function is non-differentiable, we use the right and left gradients. Let $\pi_t^+(\mathbf{y}_t, \mathbf{d}_t)$ and $\pi_t^-(\mathbf{y}_t, \mathbf{d}_t)$ be the right and the left gradients, respectively, of the value function at time t for the sample of the demand $\mathbf{d}_t$:

$$\begin{aligned}\pi_t^+(\mathbf{y}_t, \mathbf{d}_t) &= V_t^z(\mathbf{y}_t + \mathbf{1}, \mathbf{d}_t) - V_t^z(\mathbf{y}_t, \mathbf{d}_t),\\ \pi_t^-(\mathbf{y}_t, \mathbf{d}_t) &= V_t^z(\mathbf{y}_t, \mathbf{d}_t) - V_t^z(\mathbf{y}_t - \mathbf{1}, \mathbf{d}_t).\end{aligned}$$

These are random values as they depend on the demand realization. Because of the demand stochasticity, we can not observe the true gradient; therefore, we use these sample gradients to estimate it. Note that the left gradient is equal to or greater than the right gradient for any state and any demand realization, $\pi_t^-(\mathbf{y}_t, \mathbf{d}_t) \geq \pi_t^+(\mathbf{y}_t, \mathbf{d}_t)$, because of the concavity of the value function. This allows us to prove the preservation of concavity under the CAVE update (see the proof for the multistage dynamic problem in Godfrey and Powell (2002b)).

The update part of the CAVE algorithm is presented in Appendix A. For more details, we refer to the primary source, Godfrey and Powell (2001).

6.1.3 Sample gradients calculation

Because of the approximation we apply in the transshipment problem, the sample gradient each period of time is a vector of sample gradients for each location separately. In turn, the sample gradient for each location contains two parts. The first is calculated from the reward function (**??**):

$$R_{it}(y_{it}+1,d_{it}) - R_{it}(y_{it},d_{it}) = \begin{cases} p_i & \text{if } y_{it} < d_{it}, \\ -h_i & \text{otherwise,} \end{cases}$$

$$R_{it}(y_{it},d_{it}) - R_{it}(y_{it}-1,d_{it}) = \begin{cases} p_i & \text{if } y_{it} \leq d_{it}, \\ -h_i & \text{otherwise.} \end{cases}$$

The second is extracted from the maximization problem (8). Intuitively, to find marginal values having one more and one less unit of inventories that comprise the right-hand side of the maximization problem, we need to calculate the shadow prices. Positive (right-hand) and negative (left-hand) shadow prices give us the rate of change in the optimal value of the objective function for a small increase and decrease of the inventory, respectively. As we adress the linear programming problem, the dual variables should give the value of the shadow prices once we solve the problem. However, this is true only for a non-degenerate optimal solution (Bazaraa et al. (2010)). In the presence of degeneracy, variation in the right-hand side constraint might lead to an infeasible optimal basis. Therefore, we need to look for an alternative basis to be able to calculate the right values of the shadow prices. For the general linear programming problem, a way to find the "true shadow prices" is proposed by Aucamp and Steinberg (1982) and Akgul (1984). For the network problem, Powell (1989) suggests to use a shortest path algorithm, which makes calculations more efficient. The author proves that the least cost, flow augmenting (decreasing) tree from each node-location to the sink node produces the vector of positive (negative) shadow prices for the minimum-cost flow problem. Where the least cost, flow augmenting path can be calculated with a shortest path algorithm over the augmenting network.

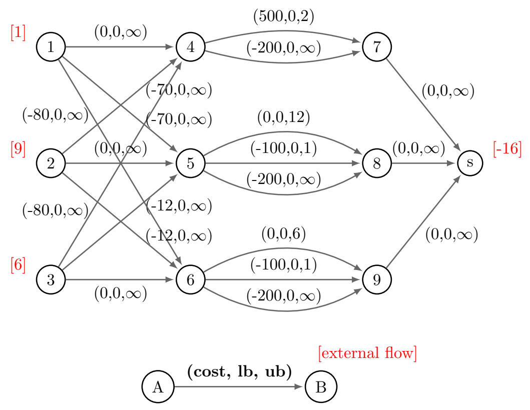
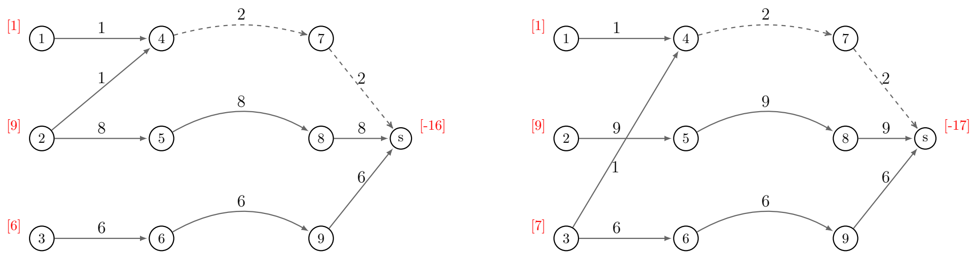


Figure 3: An example transshipment problem for 3 locations at some period t. The red number near each supply node represents the initial inventory level for each location. The arcs of the network are constrained by lower and upper bounds and have different costs. The sink node accumulates the total inventory of the whole network.

Here, we present the example of the degeneracy case for the transshipment problem. Note that for the network, degeneracy happens when one of the arcs in the optimal basis has a flow equal to the lower or the upper bounds. Fig. (3) shows the transshipment network with 3 locations. The red number near each supply node represents the initial inventory level for each location. The arcs of the network are constrained by lower and upper bounds, which each have different costs. The sink node is introduced to accumulate the total inventory of the whole network. The flow out from the sink node is equal to the sum of the initial inventory levels of all locations.

The network solver of the problem finds the following solution (Fig. (4a)). The dual variables of the supply nodes from 1 to 3 and equal 70, 0 and 0, respectively. Let us focus on location 3. The path from location 3 to the sink node contains the degenerate arc with the optimal flow equal to the upper bound. Therefore, the increase in the initial inventory level for location 3 by 1 unit will change the optimal basis. The optimal solution for an augmented problem is shown with Fig. (4b). By calculating the optimal value of the objective function for both problems, we find that the positive shadow price for location 3, $\pi^+ = -10$, differs from the dual variable. It is straightforward to check that the negative shadow price for the location 3, $\pi^- = 0$, and the shadow prices of other locations are equal to dual variables.



(a) For the initial problem. **(b)** For the augmented problem.

Figure 4: Optimal solutions.

Note that following the same procedure, in the worst case, we have to calculate $2L$ linear programming problems, while the algorithm proposed by Powell (1989) allows us to find shadow prices with 2 runs of the shortest path algorithm (one for the positive and another for the negative shadow prices).

7 Numerical experiments

In this Section, we analyze the performance of the policies presented in the paper. In the first part, we present the results of the simulation study for a small number of locations and a short time horizon. That allows us to compare heuristic methods to the optimal transshipment policy derived by the dynamic programming approach. In the second part, we focus on larger networks that represent real-life problems. We have implemented algorithms using C++ in Apple's 10.7 Environment and the experiments are conducted on a MacBook Pro 2.3 GHz Intel Core i5 processor with 4 GB RAM. For solving linear programs, we use the commercial linear programming solver Gurobi version 5.6.0.

7.1 Two-location networks

We assume two non-identical locations that fill in their stocks from the supplier at the beginning of the first period and then use these inventories to satisfy customer demands over the next four periods without any additional shipment from the supplier. The initial order-up-to level is fixed exogenously for each location. In line with the previous study conducted by Archibald et al. (2010), we consider the order-up-to level of each location to be approximately one standard deviation above the mean demand in this location scaled for the whole time horizon:

$$S_i = \left\lfloor \mu_i T + \sigma_i \sqrt{T} \right\rfloor, \tag{16}$$

where $\lfloor y \rfloor$ is the largest integer that is not greater than y. The reason is to model the configuration most favorable for transshipments when the inventories of the whole network has much lower chance of depletion compared to each location separately.

All parameters of the experiment are shown in Table 1. Here, we specify the holding cost h_i and sales price p_i per unit of goods. A transportation cost of $\$1$ per unit of distance per shipped unit of goods is considered. For this experiment, the distance between locations is fixed and can take one of two values that indicate relatively close and distant locations.

Table 1: Parameters for 2 location experiments.

Notation	Values
No. of locations, L	2
No. of periods, T	4
Holding cost per unit for location i, h_i [\$]	{8, 12, 20}
Sales price per unit for location i, p_i [\$]	{40, 80, 100}
Transshipment cost per unit per distance for location i, c_i [\$]	1
Distance between locations, ρ_{ij}	{29, 61}
Distribution of the demand	{Unif (a_i,b_i), Pois (λ_i), NegBin (r_i,p_i)}
Mean of the demand, μ_i	{0.5,1,1.5}

We use three different demand distributions to model the demand process at each location. The parameters of the distributions are shown in Table 2. The chosen distributions are intended to cover cases of low, medium and high variability of the demand. The attributes of the demand calculated from the distribution parameters presented in Table 2 are used to find the initial inventory level of each location with formula (16).

Table 2: Parameters of distributions for 2-location experiments.

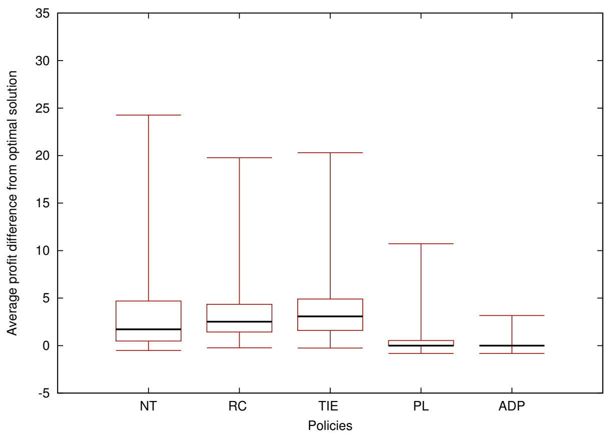
	Distributions		
	Discrete Uniform (low variance)	Poisson (medium variance)	Negative Binomial (high variance)
Notation	Unif (a_i,b_i)	Pois (λ_i)	NegBin (r_i,p_i)
Parameters	$a_i = 0$	$\lambda_i \in \{0.5, 1, 1.5\}$	$p_i = 0.8$
	$b_i \in \{1, 2, 3\}$		$r_i \in \{2, 4, 6\}$

We conduct a full factorial study based on the whole range of parameter combinations. In total, there are 4374 combinations. After eliminating symmetric combinations, we end up with 2268 scenarios. For each scenario, we perform six simulation runs (one run for each policy) with 1000 independent replications per simulation. Within each scenario, all simulation runs are applied using common random numbers. The simulation algorithm is presented in Appendix B.

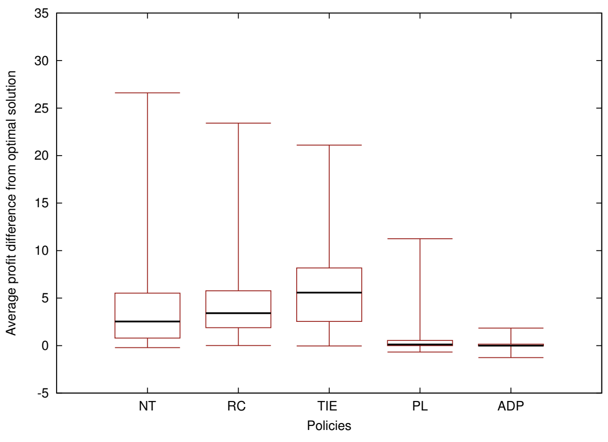
For our two-location experiment, we compare six transshipment policies: an optimal transshipment policy found by the DP procedure (DP), a no-transshipment policy (NT), reactive transshipment policy from the closest location in case of stockout (RC), a proactive transshipment policy based on the inventory equalization between locations (TIE), a lookahead policy (PL), and a proactive transshipment policy produced by the ADP procedure (ADP) with 1000 iterations to learn the value function approximation.

The statistical comparison using a paired t-test at 99% confidence interval shows that in 99% of scenarios (2254 of 2268) there is no significant difference between the ADP and DP policies, which is far greater than for the other policies. Moreover, other measures, such as the average size and frequency of transshipment over time show the similarity of the ADP and DP policies. PL demonstrates comparable results to ADP policy (in 89% of scenarios, there are no significant differences between PL and ADP policies). The overall performance of the policies compared to the optimal results obtained by the dynamic programming approach are presented in Fig. 5 for different distributions of the demand. We use the difference from the optimal solution to evaluate the results. Usually, the optimality gap is calculated as a percentage of the optimal value; however, in our case, the profit of the network contains positive and negative parts that might result in positive, negative or zero profit values and, therefore, cannot be used for the percentage calculation.

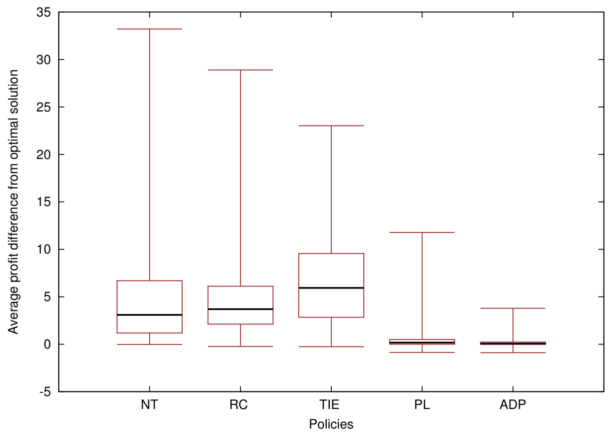
Fig. 5 supports the intuition that the advantage of a transshipment policy over a no-transshipment policy is higher for items with greater variability in demand. Moreover, the increase in demand variability indicates the drop in the performance of other policies as well. This might be the result of unnecessary movements of stocks, as policies are conducted in the anticipation of future stockouts and therefore decrease the profitability of policies. Nevertheless, ADP demonstrates consistent results compared to other policies.



(a) Unifrom.



(b) Poisson.



(c) Negative Binomial.

Figure 5: Box-and-whisker plots with median bars illustrate the distance from the optimal solution for 5 different policies (NT, RC, TIE, PL, ADP) for the case of 2 locations and various distributions of demand.

Table 3: Average profit difference from the optimal solution calculated for 5 different transshipment policies (NT, RC, TIE, PL, ADP) for the case of 2 identical locations and variable parameters.

Parameter	Value	NT vs. Opt	RC vs. Opt	TIE vs. Opt	PL vs. Opt	ADP vs. Opt
p_i	40	0.62	2.31	7.79	0.07	0.03
	80	3.09	1.41	4.65	0.24	0.14
	100	5.09	1.74	3.84	0.44	0.12
h_i	8	2.32	1.78	5.77	0.23	0.12
	12	2.77	1.81	5.51	0.22	0.09
	20	3.70	1.88	5.00	0.31	0.07
ρ_{ij}	29	4.90	1.49	2.01	0.44	0.14
	61	0.96	2.15	8.85	0.06	0.05
Unif (a_i,b_i)	$(0,1)$	0.92	0.99	1.36	0.12	0.00
	$(0,2)$	3.22	1.48	3.76	0.28	0.17
	$(0,3)$	2.79	1.60	6.82	0.47	0.09
Pois (λ_i)	0.5	1.37	1.88	4.36	0.13	0.04
	1	2.40	1.38	3.84	0.18	0.00
	1.5	4.52	2.00	9.60	0.58	0.13
NegBin (r_i,p_i)	$(2,0.8)$	1.62	2.03	4.15	0.17	0.11
	$(4,0.8)$	3.13	2.63	5.96	0.20	0.27
	$(6,0.8)$	6.43	2.41	9.00	0.13	0.10
Overall		2.93	1.82	5.43	0.25	0.09

To analyze the dependency of policy performance on a range of parameters, we focus on 162 scenarios of 2268, where locations are identical for all parameters from Table 1 except one. The results are presented in Table 3. The sample of results for a full set of scenarios is given in Appendix C. While the overall difference between the DP and NT results is not considerable for a two-location network, our results confirm previous findings of the advantage of conducting transshipments. Moreover, our study demonstrates that more advanced heuristic methods yield better results.

7.2 Multiple-location networks

For multi-location networks, we conduct two types of experiments. The first one uses a perfect foresight concerning the demand process to evaluate policies and compares the results with the optimal deterministic solution, which is obtained by solving a large-scale network flow problem. This set of experiments is conducted to analyze the effects of network size, transshipment costs and imbalance of initial inventories on transshipment policies. The second type of experiment considers that the demand is unknown and has a Poisson distribution. In this set of experiments, we compare transshipment policies for different network configurations. As industry-size stochastic dynamic problems are hard to solve to optimality, the results of these experiments are compared with no-transshipment heuristics, as often applied in practice.

Within this study, we focus on certain parameters of the multi-location problem and, therefore, fix the other parameters. The number of locations is fixed at 5, 10, 15, or 20 for each scenario. The time horizon is set to equal 28 days. The holding cost and sales price are chosen to be 5$ and 80$ per item, respectively. The distance between locations is the euclidean distance calculated at the coordinates of locations that are uniformly distributed on the grid $[0, 100] \times [0, 100]$. Other parameters are specified further for each experiment.

7.2.1 Perfect foresight as an Upper Bound

As the problem size prohibits us from calculating an optimal solution against which to compare our results, we solve the problem with perfect foresight, e.g., knowing demand in advance. This results in the deterministic demand problem (6):

$$\max_{z^d_{ijt}, z^s_{ijt}} \left(\sum_{t=0}^{T-1} \Big(-\sum_{i\in\mathcal{L}} \sum_{j\in\mathcal{L}} c\rho_{ij} \left(z^d_{ijt} + z^s_{ijt}\right) + \sum_{j\in\mathcal{L}} p_j \sum_{i\in\mathcal{L}} z^d_{ijt} - \sum_{j\in\mathcal{L}} h_j \sum_{i\in\mathcal{L}} z^s_{ijt} \Big) \right), \tag{17}$$

subject to

$$\sum_{j\in\mathcal{L}} \left(z^d_{ijt} + z^s_{ijt}\right) = \sum_{j\in\mathcal{L}} z^s_{jit-1} \quad \forall i \in \mathcal{L},\ \forall t \in \mathcal{T} : t \geq 1, \tag{18}$$

$$\sum_{j\in\mathcal{L}} \left(z^d_{ijt} + z^s_{ijt}\right) = S_i \quad \forall i \in \mathcal{L},\ t = 0, \tag{19}$$

$$\sum_{i\in\mathcal{L}} z^d_{jit} \leq d_{it} \quad \forall i \in \mathcal{L},\ \forall t \in \mathcal{T}, \tag{20}$$

$$z^d_{ijt}, z^s_{ijt} \in \mathbb{Z}_{\geq 0} \quad \forall i, j \in \mathcal{L},\ \forall t \in \mathcal{T}, \tag{21}$$

where z^d_{ijt} and z^s_{ijt} are inventories transshipped from location i to location j to be sold or stored, respectively. Therefore, the transshipment decision at each period for any pair of locations z_{ijt} is equal to the sum of two values, $z^d_{ijt} + z^s_{ijt}$.

This is an integer program, which has the structure of a min-cost flow problem with integer right-hand-side values. Therefore, the optimal solution can be obtained by solving the LP relaxation of the problem.

The optimal solution of the problem (17) provides the upper bound on the optimal value of the network profit. We compare this value with the profit obtained by following heuristics: proactive transshipment policy based on the inventory equalization between locations (TIE), lookahead policy (PL), and proactive transshipment policy produced by the ADP procedure (ADP) with 1000 iterations to learn the value function approximation. The same demand realization set is used to compare the results of policies. Nevertheless, even if the demand realization is known in advance for the whole time horizon, the heuristic methods solve the problem as if it were stochastic and, therefore, the demand each time period is drawn from the set according to the specified sequence.

We consider two different sets of experiments. In the first set, the number of locations $L \in \{5, 10, 15, 20\}$ and the transshipment costs $c \in \{0.1, 0.5, 1\}$ vary. In the second set, we vary the number of locations $L \in \{5, 10, 15, 20\}$ and the initial inventory level of locations $S \in \{697, 680, \overline{697}\}$. Calculated using (16), the initial inventory level for each location is equal to 697 units. Keeping the starting inventory vector balanced, we decrease the initial inventory level for each location and set it to 680 units. To introduce imbalance, we randomize the initial inventory level of each location such that the average inventory level among all locations is equal to 697 units. The transshipment cost is set to be \$0.5 per item per km. The set of demands is generated in advance from the Poisson distribution with an arrival rate of 24 units per day.

Table 4: Percentage of upper bound calculated by 3 transshipment policies (TIE, PL, ADP) for different transshipment costs and numbers of locations (average).

No. of locations	TIE			PL			ADP		
	\$0.1 (%)	\$0.5 (%)	\$1 (%)	\$0.1 (%)	\$0.5 (%)	\$1 (%)	\$0.1 (%)	\$0.5 (%)	\$1 (%)
5	99.49	98.02	96.20	98.38	97.93	98.28	98.26	99.28	99.35
10	99.43	97.39	94.86	98.55	97.82	97.93	98.79	98.89	99.28
15	99.57	97.07	94.16	98.89	97.84	97.83	99.19	98.86	99.21
20	99.52	97.19	94.39	98.76	97.87	97.75	98.66	98.74	99.25

Table 5: Percentage of upper bound calculated by 3 transshipment policies (TIE, PL, ADP) for different initial inventories and numbers of locations (average).

No. of locations	TIE			PL			ADP		
	S=697 (%)	S=680 (%)	$S=\overline{697}$ (%)	S=697 (%)	S=680 (%)	$S=\overline{697}$ (%)	S=697 (%)	S=680 (%)	$S=\overline{697}$ (%)
5	98.02	98.33	82.45	97.93	97.15	82.00	99.28	98.94	95.51
10	97.39	98.14	84.74	97.82	96.76	38.91	98.89	98.63	92.00
15	97.07	98.04	85.42	97.84	96.61	69.04	98.86	98.50	95.34
20	97.19	98.13	79.05	97.87	96.51	13.98	98.74	98.42	92.59

The average profit obtained by TIE, PL and ADP policies normalized by the average optimal deterministic solution for the first and second sets of experiments are shown in Tables 4 and 5, respectively. The results for ADP are more consistent among the considered transshipment policies across parameters, even if they are not the highest ones compared to other heuristics. This can be explained by the separability of the used approximation: when there is more interaction between locations caused, e.g., by the cheap transshipment costs or by highly imbalanced initial inventories, the separable approximation can be less accurate and, therefore, the performance of the policy can decrease slightly. The behavior of PL policy is similar to ADP; however, the decrease in the performance is more significant when the risk of stockout is high and more iterations between locations are required. As TIE policy aims to balance inventories between locations and does not consider any cost/price information, the policy performs well when the inventories are imbalanced and it is not expensive to transship, while if the cost of transshipment increases, the TIE policy becomes more costly.

7.2.2 Different network configurations

Table 6 presents the performance of policies compared to the no-transshipment policy for an increasing number of locations. Each estimate is aggregated over 10 configurations. The transshipment cost is equal to \$0.5 per item per km. The demand is generated from the Poisson distribution with an arrival rate of 24 units per day. The results shows that even simple reactive heuristics that conduct transshipments from the closest locations (RC) might be more profitable than a no-transshipment policy. Yet, the policy that does not consider the costs, TIE, results in less profitable decisions, and if the transshipment cost is high, it might even be unprofitable. The proactive one-step lookahead policy (PL) demonstrates considerable profitability of transshipment over a non-transshipment policy; however, the results of ADP policy that take into account the properties of the true value function are approximately 1.5 times higher than those of PL. Moreover, the CPU time of ADP presented in Table 7 for 1000 iterations is comparable to the CPU time of the PL policy or, in some cases, even less.

Table 6: Average profit difference from the no-transshipment solution calculated for 4 different policies (RC, TIE, PL, ADP).

L	RC vs. NT	TIE vs. NT	PL vs. NT	ADP vs. NT
5	115.75	93.69	240.66	490.13
10	287.82	222.78	665.42	1083.39
15	460.63	361.11	1013.02	1589.52
20	652.80	677.91	1493.89	2259.93

Table 7: Average run time of algorithms for 4 different policies (RC, TIE, PL, ADP) in sec.

L	RC	TIE	PL	ADP
5	0.48	0.53	230.20	440.13
10	1.21	1.34	689.15	862.58
15	2.48	2.69	1294.14	1388.52
20	4.36	4.61	2184.04	2059.55

8 Conclusion

In this paper, we formulate and model a multi-location, multi-period transshipment problem with lost sales in cases of stockouts. We set up a dynamic program that is solved to optimality using Bellman's equation. However, the size of the state and decision spaces makes it impossible to find the optimal policy for real-life sized problems. Therefore, we use approximate dynamic programming to find a near-optimal transshipment policy based on the properties of the value function (ADP). Our policy indicates how many units should be transshipped from which sources to which destinations to maximize the profit for the whole network. As a contribution to the existent literature on lateral transshipment, a proposed policy works with non-identical locations, can be run more than once within the replenishment period and considers a no-backorders case.

In the numerical experiments, we obtain high performance for the ADP compared to a no-transshipment policy and other known heuristics. For a smaller instances wherein the optimal solution can be found, there is no significant difference between an optimal solution and the solution found by ADP in 99% of scenarios. For real-life sized problems, ADP gives high-quality solutions that achieve 92%–99% of the upper bound, as obtained through the deterministic demand problem. The observed performance of the policy for multi-location problems with different network configurations demonstrates the capabilities of ADP and suggests its usefulness for practical problems.

With a small modification, the policy can be applied to problems with non-negligible transshipment times. Determining the optimal replenishment policy is considered a challenge for future research.

References

Agrawal, V., X. Chao, S. Seshadri. 2004. Dynamic balancing of inventory in supply chains. *European Journal of Operational Research* **159** 296–317.

Akgul, M. 1984. A note on shadow prices in linear programming. *The Journal of the Operational Research Society* **35**(5) 425–431.

Allen, S.G. 1958. Redistribution of total stock over several user locations. *Naval Research Logistics Quarterly* **5**(4) 51–59.

Archibald, T.W. 2007. Modelling replenishment and transshipment decisions in periodic review multilocation inventory systems. *Journal of the Operational Research Society* **58**(7) 948–956.

Archibald, T.W., D. Black, K.D. Glazebrook. 2009. An index heuristic for transshipment decisions in multi-location inventory systems based on a pairwise decomposition. *European Journal of Operational Research* **192** 69–78.

Archibald, T.W., D. Black, K.D. Glazebrook. 2010. The use of simple calibrations of individual locations in making transshipment decisions in a multi-location inventory network. *Journal of the Operational Research Society* **61** 294–305.

Archibald, T.W., S.A.E. Sassen, L.C. Thomas. 1997. An optimal policy for a two depot inventory problem with stock transfer. *Management Science* **43**(2) 173–183.

Aucamp, D.C., D.I. Steinberg. 1982. The computation of shadow prices in linear programing. *The Journal of the Operational Research Society* **33** 557–565.

Axsäter, S. 2003. A new decision rule for lateral transshipments in inventory systems. *Management Science* **49**(9) 1168–1179.

Banerjee, A., J. Burton, S. Banerjee. 2003. A simulation study of lateral shipments in single supplier, multiple buyers supply chain networks. *International Journal of Production Economics* **81–82** 103–114.

Bazaraa, M.S., J.J. Jarvis, H.D. Sherali. 2010. *Linear Programming and Network Flows*. John Wiley & Sons, Inc., Hoboken, New Jersey.

Bellman, R.E. 1957. *Dynamic Programming*. Princeton University Press, Princeton, NJ.

Bertrand, L.P., J.H. Bookbinder. 1998. Stock redistribution in two-echelon logistics systems. *Journal of the Operational Research Society* **49** 966–975.

Bertsekas, D.P. 2012. *Dynamic Programming and Optimal Control: approximate dynamic programming*, vol. 2. 4th ed. Athena Scientific, Belmont, MA.

Burton, J., A. Banerjee. 2005. Cost-parametric analysis of lateral transshipment policies in two-echelon supply chains. *International Journal of Production Economics* **93–94** 169–178.

Çömez, Nagihan, Kathryn E. Stecke, Metin Çakanyıldırım. 2012. Multiple in-cycle transshipments with positive delivery times. *Production and Operations Management* **21**(2) 378–395.

CSNews. 2001. Optimizing Inventory and Sales of Seasonal Products. *Industry news and trends.* Retrieved from `http://www.csnews.com/industry-news-and-trends/technology/optimizing-inventory-and-sales-seasonal-products`. Accessed October 29, 2001.

Democrat & Chronicle. 2013. Umbrellas, ponchos sell out at PGA Championship. *Sports.* Retrieved from `http://www.democratandchronicle.com/story/sports/golf/pga/2013/08/09/pga-championship-weather-rain/2634441/`. Accessed August 9, 2013.

George, A.P., W.B. Powell. 2006. Adaptive stepsizes for recursive estimation with applications in approximate dynamic programming. *Journal of Machine Learning* **65**(1) 167–198.

Glazebrook, K., C. Paterson, S. Rauscher, T. Archibald. 2014. Benefits of hybrid lateral transshipments in multi-item inventory systems under periodic replenishment. *Production and Operations Management* .

Godfrey, G.A., W.B. Powell. 2001. An adaptive, distribution-free algorithm for the newsvendor problem with censored demands, with applications to inventory and distribution. *Management Science* **47**(8) 1101–1112.

Godfrey, G.A., W.B. Powell. 2002a. An adaptive dynamic programming algorithm for dynamic fleet management, i: Single period travel times. *Transportation Science* **36**(1) 21–39.

Godfrey, G.A., W.B. Powell. 2002b. An adaptive dynamic programming algorithm for dynamic fleet management, ii: Multiperiod travel times. *Transportation Science* **35**(1) 40–54.

Grahovac, J., A. Chakravarty. 2001. Sharing and lateral transshipment of inventory in a supply chain with expensive low-demand items. *Management Science* **47**(4) 579–594.

Gross, D. 1963. *Centralized inventory control in multilocation supply systems*. In: Scarf, H.E., Gilford, D.M., Shelly, M.W. (Eds.), Multistage Inventory Models and Techniques. Stanford University Press, Stanford, California, 47–84.

Healthcare Distribution Management Association. 2009. Understanding the drivers of expired pharmaceutical returns. *HDMA Returns Task Force.* Retrieved from `http://www.nacds.org/pdfs/membership/understanding_drivers.pdf`. Accessed May, 2009.

Herer, Y.T., M. Tzur, E. Yücesan. 2006. The multi-location transshipment problem. *IIE Transactions* **38** 185–200.

Karmarkar, U.S. 1980. The multiperiod multilocation inventory problem. *Operations Research* **29**(2) 215–228.

Krishnan, K., V. Rao. 1965. Inventory control in N warehouses. *Journal of Industrial Engineering* **16**(3) 212–215.

Lee, H.L. 1987. A multi-echelon inventory model for repairable items with emergency lateral transshipments. *Management Science* **33**(10) 1302–1316.

Mei-hsing, S. 2011. Can't beat the heat: Air conditioner shortages in China. *Want China Times. Economy.* Retrieved from `http://www.wantchinatimes.com/news-subclass-cnt.aspx?id=20110806000057&cid=1102`. Accessed August 6, 2011.

Minner, S., E.A. Silver, D.J. Robb. 2003. An improved heuristic for deciding on emergency transshipments. *European Journal of Operational Research* **148** 384–400.

Pastor, K. 2014. This Winter, a Good Boot Is Hard to Find. *The New York Times.* Retrieved from `http://www.nytimes.com/2014/02/10/nyregion/this-winter-a-good-boot-is-hard-to-find.html?_r=0`. Accessed February 09, 2014.

Paterson, C., G. Kiesmüller, R. Teunter, K.D. Glazebrook. 2011. Inventory models with lateral transshipments: A review. *European Journal of Operational Research* **210** 125–136.

Paterson, C., R. Teunter, K.D. Glazebrook. 2012. Enhanced lateral transshipments in a multi-location inventory system. *European Journal of Operational Research* **210** 125–136.

Powell, W.B. 1989. A review of sensitivity results for linear networks and a new approximation to reduce the effects of degeneracy. *Transportation Science* **23**(4) 231–243.

Powell, W.B. 2011. *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. 2nd ed. Wiley, Hoboken, New Jersey.

Robinson, L.W. 1990. Optimal an approximate policies in multiperiod, multilocation inventory models with transshipments. *Operations Research* **38**(2) 278–295.

Rockafeller, R.T. 1996. *Convex Analysis*. 2nd ed. Princeton University Press, Princeton, NJ.

Seidscher, A., S. Minner. 2013. A semi-markov decision problem for proactive and reactive transshipments between multiple warehouses. *European Journal of Operational Research* **230** 42–52.

Sutton, R.S., A.G. Barto. 1998. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA.

Tiacci, L., S. Saetta. 2011. A heuristic for balancing the inventory level of different locations through lateral shipments. *International Journal of Production Economics* **131** 87–95.

US Food and Drug Administration. 2014. First Annual Report on Drug Shortages for Calendar Year 2013. *Department of Health and Human Services.* Retrieved from `http://www.fda.gov/downloads/drugs/drugsafety/drugshortages/ucm384892.pdf`. Accessed February 5, 2014.

Van der Heide, G., K.J. Roodbergen. 2013. Transshipment and rebalancing policies for library books. *European Journal of Operational Research* **228** 447–456.

Wong, H., G.J. van Houtum, D. Cattrysse, D. van Oudheusden. 2005. Stocking decisions for repairable spare parts pooling in a multi-hub system. *International Journal of Production Economics* **93–93** 309–317.

Wong, H., G.J. van Houtum, D. Cattrysse, D. van Oudheusden. 2006. Multi-item spare parts systems with lateral transshipments and waiting time constraints. *European Journal of Operational Research* **171** 1071–1093.

Zipkin, P. 2008. On the structure of lost-sales inventory models. *Operations Research* **56**(4) 937–944.

Appendix A. Approximate Dynamic Programming

Algorithm 1: Forward ADP algorithm around post-decision state

```
Input  : $L$, $T$, $c$, $N$, $\alpha$,$\epsilon$,
         $S_i$, $h_i$, $p_i$, $\rho_{ij}$ for $i, j \in \mathcal{L}$;
Output : $\overline{V}_t^N(Y)$ for $\forall\ t \in \mathcal{T}$;
begin
    initialize $\overline{V}_t^0(Y) = 0$ for $\forall\ t \in \mathcal{T}$;
    for $n = 1, \dots, N$ do
        initialize $x_{i0} = S_i$ for $\forall\ i \in \mathcal{L}$;
        for $t = 0, \dots, T-1$ do
            solve the maximization problem:
                $$\hat{v}_t = \max_{\mathbf{z}_t} \Big( - C_t(\mathbf{x}_t, \mathbf{z}_t) + \overline{V}_t^{n-1}(\mathbf{y}_t) \Big)$$
            if $t > 0$ then
                update approximation $\overline{V}_{t-1}^{n-1}(\mathbf{y_{t-1}})$:
                    $$\overline{V}_{t-1}^{n}(\mathbf{y}_{t-1}) = (1-\alpha)\overline{V}_{t-1}^{n-1}(\mathbf{y}_{t-1}) + \alpha\hat{v}_t$$
            if Uniform(0,1) $> \epsilon$ then
                calculate post-decision state $\mathbf{y}_t$:
                    $$\mathbf{y}_t = X^{M,z}(\mathbf{x}_t, \mathbf{z}_t)$$
            else
                take next post-decision state $\mathbf{y}_t$ randomly;
            generate random demand vector $\mathbf{d}_t$;
            calculate next pre-decision state $\mathbf{x}_{t+1}$:
                $$\mathbf{x}_{t+1} = X^{M,d}(\mathbf{y}_t, \mathbf{d}_t)$$
```

Algorithm 2: CAVE update of the approximation

```
Input  : $L$, $\mathbf{x}_t$, $\delta^+$, $\delta^-$, $\alpha$, $\pi_t^+$, $\pi_t^-$, $K_t^n$, $v_t^n$, $u_t^n$;
Output : $K_t^{n+1}$, $v_t^{n+1}$, $u_t^{n+1}$ ;
begin
    for $i = 1, \dots, L$ do
        $k^- = \min\{k \in K_{it}^n : v_{it}^{k,n} \le (1-\alpha)v_{it}^{k+1,n} + \alpha\pi_{it}^-\}$,
        $k^+ = \max\{k \in K_{it}^n : (1-\alpha)v_{it}^{k-1,n} + \alpha\pi_{it}^+ \le v_{it}^{k,n}\}$,
        Define the smoothing interval:
            $\Delta = [\min\{x_{it} - \delta^-, u_{it}^{k^-,n}\}, \max\{x_{it} + \delta^+, u_{it}^{k^+ +1,n}\}]$
        Create new breakpoints at $x_{it}$ and the end points of $\Delta$; add them to $u_t^{n+1}$;
        Add according indexes to $K_{it}^{n+1}$;
        For each segment in the interval update the slope:
            $v_{it}^{k,n+1} = \alpha\pi_{it} + (1-\alpha)v_{it}^{k,n}, \quad i \in \{1, \dots, L\}$,
        where
            $\pi_{it} = \begin{cases} \pi_{it}^- & \text{if } u_{it}^{k,n} < x_{it}, \\ \pi_{it}^+ & \text{otherwise.} \end{cases}$
```

Appendix B. Simulation algorithm

Algorithm 3: Evaluation of a policy

```
Input  : $L$, $T$, $c$, $N$, policy $\pi$,
         $S_i$, $h_i$, $p_i$, $\rho_{ij}$ for $i, j \in \mathcal{L}$;
Output : Average profit after $N$ iterations;
begin
    for $n = 1, \dots, N$ do
        initialize $x_{i0} = S_i$ for $\forall\ i \in \mathcal{L}$;
        for $t = 0, \dots, T-1$ do
            calculate transshipments between location $\mathbf{z}_t^\pi$;
            get transshipment cost $C_t(\mathbf{x}_t, \mathbf{z}_t^\pi)$;
            generate random demand vector $\mathbf{d}_t$;
            get reward $R_t(\mathbf{x}_t, \mathbf{d}_t, \mathbf{z}_t^\pi)$;
            calculate next state $\mathbf{x}_{t+1}$;
            get profit $P_t = (R_t(\mathbf{x}_t, \mathbf{d}_t, \mathbf{z}_t^\pi) - C_t(\mathbf{x}_t, \mathbf{z}_t^\pi))$;
        get total profit $P = \sum_{t=0}^{T-1} P_t$
    get average profit $P/N$
```

Appendix C. Results

Table 8: Two location results: $p_1 = 40, p_2 = 80, h_1 = 8, h_2 = 12$ (nonidentical locations)

Distance, $\rho_i j$	Distrib $i = 1$	Distrib $j = 2$	Opt	NT	RC	TIE	PL	ADP
29	Unif(0,1)	Unif(0,1)	95.07	93.97	94.28	94.77	95.07	95.07
61	Unif(0,1)	Unif(0,1)	93.97	93.97	92.93	90.51	93.97	93.97
29	Unif(0,2)	Unif(0,1)	205.98	201.26	201.49	203.79	205.98	205.98
29	Unif(0,1)	Unif(0,2)	143.97	142.51	142.80	143.37	143.97	143.97
61	Unif(0,2)	Unif(0,1)	201.26	201.26	199.15	198.85	201.26	201.26
61	Unif(0,1)	Unif(0,2)	142.51	142.51	140.46	138.44	142.51	142.51
29	Unif(0,3)	Unif(0,1)	284.15	282.25	280.79	283.39	283.63	284.27
29	Unif(0,1)	Unif(0,3)	169.75	169.16	168.60	169.40	169.75	169.75
61	Unif(0,3)	Unif(0,1)	282.72	282.25	276.66	277.66	282.72	282.72
61	Unif(0,1)	Unif(0,3)	169.16	169.16	164.46	163.66	169.16	169.16
29	Unif(0,2)	Unif(0,2)	252.51	247.82	250.33	249.62	252.30	252.51
61	Unif(0,2)	Unif(0,2)	247.82	247.82	246.39	241.78	247.82	247.82
29	Unif(0,3)	Unif(0,2)	331.63	328.82	329.42	329.99	331.16	331.63
29	Unif(0,2)	Unif(0,3)	279.09	275.24	277.44	276.68	279.09	279.03
61	Unif(0,3)	Unif(0,2)	329.01	328.82	324.77	321.91	329.01	329.01
61	Unif(0,2)	Unif(0,3)	275.24	275.24	272.80	268.61	275.24	275.24
29	Unif(0,3)	Unif(0,3)	360.80	357.58	359.54	357.36	360.21	360.83
61	Unif(0,3)	Unif(0,3)	357.92	357.58	355.76	344.73	357.92	357.92
29	Pois(0.5)	Pois(0.5)	75.91	74.13	74.78	74.02	75.58	75.91
61	Pois(0.5)	Pois(0.5)	73.93	74.13	70.87	65.16	74.13	73.93
29	Pois(1)	Pois(0.5)	156.92	154.58	153.57	154.53	156.64	156.92
29	Pois(0.5)	Pois(1)	100.20	98.94	99.50	98.91	100.19	100.13
61	Pois(1)	Pois(0.5)	154.65	154.58	147.27	146.21	154.65	154.65
61	Pois(0.5)	Pois(1)	99.03	98.94	93.20	90.59	98.94	99.03
29	Pois(1.5)	Pois(0.5)	271.70	267.20	264.66	268.37	270.20	271.70
29	Pois(0.5)	Pois(1.5)	150.48	149.04	148.63	148.92	150.60	150.48
61	Pois(1.5)	Pois(0.5)	267.90	267.20	256.94	258.99	267.90	267.92
61	Pois(0.5)	Pois(1.5)	149.01	149.04	140.92	139.54	149.00	149.01
29	Pois(1)	Pois(1)	184.84	182.04	183.95	182.73	184.58	184.92
61	Pois(1))	Pois(1)	182.16	182.04	179.40	174.47	182.12	182.12
29	Pois(1.5)	Pois(1)	299.59	294.65	296.92	297.50	298.50	299.53
29	Pois(1)	Pois(1.5)	233.98	231.19	233.34	231.73	234.17	233.98
61	Pois(1.5))	Pois(1)	295.43	294.65	291.80	287.10	295.42	295.36
61	Pois(1)	Pois(1.5)	231.26	231.19	228.22	221.33	231.39	231.41
29	Pois(1.5)	Pois(1.5)	349.99	344.83	348.41	344.58	348.84	349.57
61	Pois(1.5)	Pois(1.5)	345.64	344.83	342.33	328.32	345.64	345.55
29	NegBin(2,0.8)	NegBin(2,0.8)	53.10	51.42	52.25	51.34	52.85	53.10
61	NegBin(2,0.8)	NegBin(2,0.8)	51.42	51.42	47.80	41.68	51.42	51.42
29	NegBin(4,0.8)	NegBin(2,0.8)	130.64	127.28	127.70	128.10	130.29	130.48
29	NegBin(2,0.8)	NegBin(4,0.8)	78.04	75.48	78.04	77.39	77.98	78.04
61	NegBin(2,0.8)	NegBin(2,0.8)	127.49	127.28	120.40	119.10	127.50	127.52
61	NegBin(2,0.8)	NegBin(4,0.8)	75.95	75.48	70.74	68.40	75.48	75.48
29	NegBin(6,0.8)	NegBin(2,0.8)	247.83	243.95	241.26	245.20	247.75	247.60
29	NegBin(2,0.8)	NegBin(6,0.8)	130.08	127.65	127.09	127.87	129.50	130.06
61	NegBin(6,0.8)	NegBin(2,0.8)	244.76	243.95	232.90	234.80	244.76	244.49
61	NegBin(2,0.8)	NegBin(6,0.8)	127.69	127.65	118.74	117.47	127.65	127.65
29	NegBin(4,0.8)	NegBin(4,0.8)	153.19	149.66	151.95	150.66	153.16	153.06
61	NegBin(4,0.8)	NegBin(4,0.8)	149.87	149.66	146.03	140.26	150.02	149.91
29	NegBin(6,0.8)	NegBin(4,0.8)	272.54	266.34	269.26	269.77	272.27	272.44
29	NegBin(4,0.8)	NegBin(6,0.8)	203.37	200.58	202.81	199.98	202.72	203.50
61	NegBin(6,0.8)	NegBin(4,0.8)	267.45	266.34	263.11	256.26	267.47	267.47
61	NegBin(4,0.8)	NegBin(6,0.8)	200.85	200.58	196.66	186.47	200.85	200.68
29	NegBin((6,0.8)	NegBin((6,0.8)	321.36	314.67	320.00	317.42	320.99	321.39
61	NegBin((6,0.8)	NegBin((6,0.8)	316.08	314.67	312.58	298.48	316.08	315.71